Základy programování v C++ - 3. cvičení Podmínky a cykly (while)

Zuzana Petříčková

2. října 2019

Už jsme probírali

- čtení textu ze systémové konzole a jeho výpis na konzoli
  - cin, cout,...
- proměnné a datové typy
  - číselná aritmetika, operátory, knihovna **cmath**
  - implicitní a explicitní přetypování
- Příkazy pro řízení běhu programu
  - podmíněné bloky (podmínky)
- Dobrovolný domácí úkol a jeho řešení

Řízení běhu programu

- **podmíněné bloky (podmínky)**
- cykly (smyčky)
- podprogramy (funkce)

Řízení běhu programu - Podmíněné bloky

Jeden příkaz:

```
if (testovaci_podminka)
 prikaz1;
```

Více příkazů:

```
if (testovaci_podminka)
{
      prikaz1;
      prikaz2;
      ...
}
```

testovaci_podminka = výraz, který lze převést na logickou hodnotu

Řízení běhu programu - Podmíněné bloky

IF - ELSE:

```
if (testovaci_podminka)
        prikaz1;
else
        prikaz2;
```

Více příkazů:

```
if (testovaci_podminka)
        prikaz1;
else
{
        prikaz2;
        prikaz3;
        ...
}
```

Řízení běhu programu - Podmíněné bloky

IF - ELSE IF - ELSE:

```
if (testovaci_podminka1)
      prikaz1;
else if (testovaci_podminka2)
      prikaz2;
else
      prikaz3;
```

- opět lze použít i bloky příkazů

Řízení běhu programu - Podmíněné bloky

Testovací podmínky

- boolovský výraz
- libobolný výraz, který lze převézt na typ **bool** (čísla, znaky, ukazatele)

Relační operátory (operátory pro porovnání)

- == ... rovnost (NE pro racionální čísla)
- != ... nerovnost (NE pro racionální čísla)
- < ... je menší
- <= ... je menší nebo rovno
- > ... je větší
- >= ... je větší nebo rovno

Řízení běhu programu - Podmíněné bloky

Relační operátory (operátory pro porovnání)

- == ... rovnost (NE pro racionální čísla)
- != ... nerovnost (NE pro racionální čísla)
- < ... je menší
- <= ... je menší nebo rovno
- > ... je větší
- >= ... je větší nebo rovno

Příklad

```
int a = 1000, b = 2000;
bool vysl = a > b;
vysl = a;
cout << ( a <= b ) << endl;
if (a == b)
    cout << "Cisla se rovnaji.";
else
    cout << "Cisla se nerovnaji.";
```

Řízení běhu programu - Podmíněné bloky

Logické operátory (operátory pro porovnání)

- && ... logické AND
- || ... logické OR
- ! ... logická negace

Cvičení (papír a tužka)

```
! (1 || 0)
! (1 || 1 && 0 )
( (1 && 0) || ! 0 )
```

Příklad

```
if (a == 0 || b == 0)
    cout << "Jedno z cisel je 0.";
```

Číselné, relační a logické operátory – priorita a asociativita

Priorita	Operátor	Asociativita
2	postfixové ++ −−	zleva
3	prefixové ++ −−	zprava
	unární + −	zprava
	!	zprava
5	* / %	zleva
6	+ −	zleva
8	< <= > >=	zleva
9	'==' '! ='	zleva
13	&&	zleva
14	||	zleva
15	=	zprava
	'+=' '-=' '*=' '/=' '%='	zprava

Řízení běhu programu - Podmíněné bloky

Příklad: složitější podmínka

```
...
int main()
{
    int cis;
    int minimum = 0, maximum = 2000;
    cout << "Zadej cislo" << endl;
    cin >> cis;
    if (( cis < minimum ) || (cis > maximum ))
    {
        cout << "Cislo lezi mimo pozadovany interval!" << endl;
        return 1;
     }
    ...
    return 0;
}
```

Řízení běhu programu - Podmíněné bloky

Zkrácené vyhodnocování podmínek

```
    ...
    int x, y;
    cout << "Zadej cisla: ";
    cin >> x;
    cin >> y;
    if ((x != 0) && (y % x == 0))
        cout << "Cislo x je delitelem cisla y" << endl;
    ...
}
```

- **konjunkce:** je–li první podmínka nepravda, je celý výrok nepravda (druhá podmínka se nevyhodnocuje)
- **disjunkce:** je–li první podmínka pravda, je celý výrok pravda (druhá podmínka se nevyhodnocuje)

Řízení běhu programu - Podmíněné bloky

Automatická konverze na logickou hodnotu

- nenulové číslo → 1 (true)
- nula → 0 (false)

```
...
int main()
{
    int cislo;
    cout << "Zadej cislo" << endl;
    cin >> cislo;
    bool b = cislo;
    ...
    if (!cislo) // if (cislo == 0)
        cout << "Zadal jsi nulu." << endl;
    return 0;
}
```

Řízení běhu programu - Podmíněné bloky

Automatická konverze na logickou hodnotu

- nenulové číslo → 1 (true)
- nula → 0 (false)

```
...
int main()
{
    int cislo;
    cout << "Zadej cislo" << endl;
    cin >> cislo;
    bool b = cislo;
    ...
    if (cislo)    // if (cislo !=0)
        ;         // prazdny prikaz
    else
        cout << "Zadal jsi nulu." << endl;
    return 0;
}
```

Řízení běhu programu - Cykly

Cykly:

- část kódu, která se provádí vícekrát
- o dalším opakování rozhoduje splnění podmínky
- počet opakování může a nemusí být znám předem

Řízení běhu programu - Cyklus WHILE

Jeden příkaz:

```
while (testovaci_podminka)
    prikaz1;
```

Více příkazů:

```
while (testovaci_podminka)
{
      prikaz1;
      prikaz2;
      ...
}
```

testovaci_podminka = výraz, který lze převést na logickou hodnotu

Řízení běhu programu - Cyklus WHILE

Příklad 1:

```
...
int i = 0;
while (i <= 10)
{
    cout << i << " " << i*i << endl;
    i++;
}
...
```

Příklad 2: faktoriál
program, který pro zadané celé číslo n vypíše $n!$

Řízení běhu programu - Cyklus WHILE

Příklad: faktoriál

```
{
    int n;
    cout << "Zadej hodnotu, z ktere chces spocitat faktorial:
    cin >> n;
    long long y = 1;
    while (n > 0)
    {
        y *= n;
        n--;
    }
    cout << "Faktorial je " << y <<  endl;
}
```

Řízení běhu programu - Cyklus DO-WHILE

```
do
{
        prikaz1;
        prikaz2;
        ...
}
while (testovaci_podminka)
```

Řízení běhu programu - Cyklus DO-WHILE

Příklad 1: až naprší ...

```
int main()
{
    char odpoved;
    do {
        cout << "Naprselo? (A/N)" << endl;
        cin >> odpoved;
    }
    while(odpoved != 'A');
    cout << "Tak uschne!" << endl;
    return 0;
}
```

Příklad 2: program, který načte číslo s daného intervalu

Řízení běhu programu - Cyklus DO-WHILE

Příklad 2: program, který načte číslo s daného intervalu

```
{
    int x,   min = 0, max = 100;;
    do {
        cout << "Zadej cislo z intervalu <" <<
          min << "," <<  max << ">" << endl;
        cin >> x;
        if (!cin) //  kontrola vstupu
        {
            cin.clear();
            cin.ignore(256,'\n');
            x = min - 1;
        }
    } while ((x > max) || (x < min));
        cout << "Trefa! " << endl;
}
```

Problém racionálních čísel

Příklad

- Kód, který v cyklu WHILE přičítá k proměnné f typu double inicializované na 0 hodnotu 0.1. Cyklus ukončí, jakmile f bude rovno 1.

```
...
double f = 0.0;
while(f != 1.0)
{
    f += 0.1;
}
...
```

→ zacyklí se

- Racionální čísla jsou na počítači reprezentována nepřesně → místo operátorů -="a "!="testovat, zda jsou čísla dostatečně blízko.

Problém racionálních čísel

Příklad

- Kód, který v cyklu WHILE přičítá k proměnné f typu double inicializované na 0 hodnotu 0.1. Cyklus ukončí, jakmile f bude rovno 1.

```
...
double eps = 1e-5;
...
double f = 0.0;
while(abs(f-1.0) > eps)
{
  f += 0.1;
}
...
```

Příklady na procvičení IF-ELSE a WHILE:

1. Napište program, který pro zadané celé číslo rozhodne, zda je sudé (a vypíše výsledek).
2. Napište program, který porovná dvě zadaná celá čísla a vypíše, které je větší.
3. Napište program, který porovná tři zadaná celá čísla a vypíše, které je největší.
4. (kdo stihne) Napište program, který vypíše tři zadaná celá čísla v pořadí od největšího po nejmenší.
5. **mocnina:** Napište program, který pro zadané x a n (double x, int n) spočte a vypíše x^n.